



# Red Hat Data Grid 8.1

## Data Grid コマンドラインインターフェイス

CLI を使用してデータにアクセスし、Data Grid を管理します



# Red Hat Data Grid 8.1 Data Grid コマンドラインインターフェイス

---

CLI を使用してデータにアクセスし、Data Grid を管理します

## 法律上の通知

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 概要

コマンドラインインターフェイス (CLI) から Data Grid サーバーに接続して、データにアクセスし、管理操作を実行します。

## 目次

RED HAT DATA GRID .....	4
DATA GRID のドキュメント .....	5
DATA GRID のダウンロード .....	6
多様性を受け入れるオープンソースの強化 .....	7
<b>第1章 DATA GRID CLI のスタートガイド .....</b>	<b>8</b>
1.1. ユーザーの作成と変更 .....	8
1.2. DATA GRID SERVER への接続 .....	8
1.3. CLI リソースのナビゲート .....	9
1.4. DATA GRID SERVER のシャットダウン .....	11
<b>第2章 PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI .....</b>	<b>13</b>
2.1. DATA GRID コマンドラインインターフェイス (CLI) を使用したキャッシュの作成 .....	13
2.2. キャッシュエントリーの追加 .....	14
2.3. キャッシュのクリアとエントリーの削除 .....	15
2.4. キャッシュの削除 .....	15
<b>第3章 バッチ操作の実行 .....</b>	<b>16</b>
3.1. ファイルを使用したバッチ操作の実行 .....	16
3.2. インタラクティブなバッチ操作の実行 .....	16
<b>第4章 DATA GRID CLI の設定 .....</b>	<b>18</b>
4.1. DATA GRID CLI プロパティと永続ストレージの設定 .....	18
4.2. コマンドエイリアスの作成 .....	18
4.3. DATA GRID SERVER 接続の信頼 .....	19
4.4. DATA GRID CLI ストレージディレクトリー .....	19
<b>第5章 カウンターの操作 .....</b>	<b>21</b>
5.1. カウンターの作成 .....	21
5.2. カウンターへのデルタの追加 .....	22
<b>第6章 PROTOBUF メタデータを使用したキャッシュのクエリー .....</b>	<b>23</b>
6.1. メディアタイプの設定 .....	23
6.2. PROTOBUF スキーマの登録 .....	24
6.3. PROTOBUF スキーマを使用したキャッシュのクエリー .....	25
<b>第7章 クロスサイトレプリケーション操作の実行 .....</b>	<b>28</b>
7.1. バックアップ場所のオフラインおよびオンライン化 .....	28
7.2. バックアップ場所への状態のプッシュ .....	28
<b>第8章 コマンドリファレンス .....</b>	<b>30</b>
8.1. ADD(1) .....	30
8.2. ALIAS(1) .....	30
8.3. CACHE(1) .....	31
8.4. CAS(1) .....	31
8.5. CD(1) .....	32
8.6. CLEARCACHE(1) .....	32
8.7. CONFIG(1) .....	33
8.8. CONNECT(1) .....	34
8.9. CONTAINER(1) .....	35
8.10. COUNTER(1) .....	35

8.11. CREATE(1)	35
8.12. DESCRIBE(1)	36
8.13. DISCONNECT(1)	37
8.14. DROP(1)	37
8.15. ENCODING(1)	38
8.16. GET(1)	38
8.17. HELP(1)	39
8.18. LOGGING(1)	39
8.19. LS(1)	40
8.20. MIGRATE(1)	40
8.21. PATCH(1)	41
8.22. PUT(1)	43
8.23. QUERY(1)	44
8.24. QUIT(1)	44
8.25. REMOVE(1)	45
8.26. RESET(1)	45
8.27. SCHEMA(1)	46
8.28. SHUTDOWN(1)	46
8.29. SITE(1)	47
8.30. STATS(1)	48
8.31. TASK(1)	48
8.32. UNALIAS(1)	49
8.33. USER(1)	49
8.34. VERSION(1)	51



## RED HAT DATA GRID

Data Grid は、高性能の分散型インメモリーデータストアです。

### スキーマレスデータ構造

さまざまなオブジェクトをキーと値のペアとして格納する柔軟性があります。

### グリッドベースのデータストレージ

クラスター間でデータを分散および複製するように設計されています。

### エラスティックスケールリング

サービスを中断することなく、ノードの数を動的に調整して要件を満たします。

### データの相互運用性

さまざまなエンドポイントからグリッド内のデータを保存、取得、およびクエリーします。



## DATA GRID のドキュメント

Data Grid のドキュメントは、Red Hat カスタマーポータルで入手できます。

- [Data Grid 8.1 ドキュメント](#)
- [Data Grid 8.1 コンポーネントの詳細](#)
- [Data Grid 8.1 でサポートされる設定](#)
- [Data Grid 8 機能のサポート](#)
- [Data Grid で非推奨の機能](#)

## DATA GRID のダウンロード

Red Hat カスタマーポータルで [Data Grid Software Downloads](#) にアクセスします。



### 注記

Data Grid ソフトウェアにアクセスしてダウンロードするには、Red Hat アカウントが必要です。

## 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) をご覧ください。

## 第1章 DATA GRID CLI のスタートガイド

コマンドラインインターフェイス (CLI) を使用すると、Data Grid サーバーにリモートで接続して、データにアクセスし、管理機能を実行できます。

### 1.1. ユーザーの作成と変更

Data Grid Server では、デフォルトのプロパティレームに対してユーザーを認証する必要があります。Data Grid Server にアクセスする前に、少なくとも1人のユーザーとパスワードを作成して認証情報を追加する必要があります。ユーザーが属するセキュリティー承認グループを追加および変更することもできます。

#### 手順

1. `$RHDG_HOME` でターミナルを開きます。
2. `user` コマンドを使用して Data Grid ユーザーを作成および変更します。

#### ヒント

このコマンドの使用の詳細は、`help user` を実行してください。

#### ユーザーとパスワードの作成

- Linux

```
$ bin/cli.sh user create myuser -p "qwer1234!"
```

- Microsoft Windows

```
$ bin\cli.bat user create myuser -p "qwer1234!"
```

#### グループメンバーシップを持つユーザーの作成

- Linux

```
$ bin/cli.sh user create myuser -p "qwer1234!" -g supervisor,reader,writer
```

- Microsoft Windows

```
$ bin\cli.bat user create myuser -p "qwer1234!" -g supervisor,reader,writer
```

### 1.2. DATA GRID SERVER への接続

Data Grid への CLI 接続を確立します。

#### 前提条件

ユーザーの認証情報を追加し、稼働中の Data Grid Server インスタンスが1つ以上ある。

#### 手順

1. `$RHDG_HOME` でターミナルを開きます。

2. CLI を起動します。

- Linux:

```
$ bin/cli.sh
```

- Microsoft Windows:

```
$ bin\cli.bat
```

3. `connect` コマンドを実行し、プロンプトが表示されたらユーザー名とパスワードを入力します。

- 11222 のデフォルトポート上の Data Grid Server:

```
[disconnected]> connect
```

- ポートオフセットが 100 の Data Grid Server:

```
[disconnected]> connect 127.0.0.1:11322
```

### 1.3. CLI リソースのナビゲート

Data Grid CLI は、Data Grid クラスターリソースのリスト表示、説明、および操作を可能にするナビゲート可能なツリーを公開します。

#### ヒント

Tab キーを押して、使用可能なコマンドとオプションを表示します。`-h` オプションを使用して、ヘルプテキストを表示します。

Data Grid クラスターに接続すると、デフォルトのキャッシュコンテナのコンテキストで開きます。

```
[//containers/default]>
```

- `ls` を使用してリソースをリストします。

```
[//containers/default]> ls
caches
counters
configurations
schemas
tasks
```

- `cd` を使用してリソースツリーをナビゲートします。

```
[//containers/default]> cd caches
```

- `describe` を使用して、リソースに関する情報を表示します。

```
[//containers/default]> describe
{
  "name" : "default",
  "version" : "xx.x.x-FINAL",
  "cluster_name" : "cluster",
  "coordinator" : true,
  "cache_configuration_names" : [ "org.infinispan.REPL_ASYNC", "__protobuf_metadata",
  "org.infinispan.DIST_SYNC", "org.infinispan.LOCAL", "org.infinispan.INVALIDATION_SYNC",
  "org.infinispan.REPL_SYNC", "org.infinispan.SCATTERED_SYNC",
  "org.infinispan.INVALIDATION_ASYNC", "org.infinispan.DIST_ASYNC" ],
  "physical_addresses" : "[192.0.2.0:7800]",
  "coordinator_address" : "<hostname>",
  "cache_manager_status" : "RUNNING",
  "created_cache_count" : "1",
  "running_cache_count" : "1",
  "node_address" : "<hostname>",
  "cluster_members" : [ "<hostname1>", "<hostname2>" ],
  "cluster_members_physical_addresses" : [ "192.0.2.0:7800", "192.0.2.0:7801" ],
  "cluster_size" : 2,
  "defined_caches" : [ {
    "name" : "mycache",
    "started" : true
  }, {
    "name" : "__protobuf_metadata",
    "started" : true
  } ]
}
```

### 1.3.1. CLI リソース

Data Grid CLI は、以下の目的でさまざまなリソースを公開します。

- ローカルキャッシュまたはクラスター化キャッシュを作成、変更、および管理します。
- Data Grid クラスターの管理操作を実行します。

#### キャッシュリソース

```
[//containers/default]> ls
caches
counters
configurations
schemas
```

##### caches

Data Grid キャッシュインスタンス。デフォルトのキャッシュコンテナは空です。CLI を使用して、テンプレートまたは **infinispan.xml** ファイルからキャッシュを作成します。

##### counters

オブジェクトの数を記録する **Strong** カウンターまたは **Weak** カウンター。

##### configurations

Data Grid 設定。

##### schemas

キャッシュ内のデータを構造化する Protocol Buffers (Protobuf) スキーマ。

## tasks

Data Grid キャッシュ定義を作成および管理するリモートタスク。

## クラスターリソース

```
[hostname@cluster/]> ls
containers
cluster
server
```

## containers

Data Grid クラスター上のキャッシュコンテナ。

## cluster

クラスターに参加している Data Grid サーバーをリスト表示します。

## server

Data Grid サーバーを管理および監視するためのリソース。

## 1.4. DATA GRID SERVER のシャットダウン

個別に実行中のサーバーを停止するか、クラスターを正常に停止します。

### 手順

1. Data Grid への CLI 接続を作成します。
2. 次のいずれかの方法で Data Grid Server をシャットダウンします。
  - **shutdown cluster** コマンドを使用して、クラスターのすべてのノードを停止します。以下に例を示します。

```
[//containers/default]> shutdown cluster
```

このコマンドは、クラスターの各ノードの **data** フォルダにクラスターの状態を保存します。キャッシュストアを使用する場合、**shutdown cluster** コマンドはキャッシュのすべてのデータも永続化します。

- **shutdown server** コマンドおよびサーバーのホスト名を使用して、個々のサーバーインスタンスを停止します。以下に例を示します。

```
[//containers/default]> shutdown server <my_server01>
```



### 重要

**shutdown server** コマンドは、リバランス操作が完了するまで待機しません。これにより、同時に複数のホスト名を指定すると、データが失われる可能性があります。

### ヒント

このコマンドの使用法の詳細については、**help shutdown** を実行してください。

## 検証

Data Grid は、サーバーをシャットダウンしたときに以下のメッセージをログに記録します。

```
ISPN080002: Data Grid Server stopping
ISPN000080: Disconnecting JGroups channel cluster
ISPN000390: Persisted state, version=<$version> timestamp=YYYY-MM-DDTHH:MM:SS
ISPN080003: Data Grid Server stopped
```

### 1.4.1. Data Grid クラスターの再起動

シャットダウン後に Data Grid クラスターをオンラインに戻す場合、クラスターが利用できるのを待ってから、ノードの追加または削除、またはクラスター状態の変更を行う必要があります。

**shutdown server** コマンドでクラスター化ノードをシャットダウンする場合は、各サーバーを逆の順序で再起動する必要があります。

たとえば、**server1** をシャットダウンしてから、**server2** をシャットダウンする場合は、最初に **server2** を起動してから **server1** を起動する必要があります。

**shutdown cluster** コマンドでクラスターをシャットダウンすると、すべてのノードが再度参加した後にのみ、クラスターは完全に機能するようになります。

ノードは任意の順序で再起動できますが、シャットダウン前に参加していたすべてのノードが実行されるまで、クラスターは DEGRADED 状態のままになります。



## 第2章 PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI

コマンドラインインターフェイス (CLI) を使用すると、Data Grid サーバーにリモートで接続して、データにアクセスし、管理機能を実行できます。

### 2.1. DATA GRID コマンドラインインターフェイス (CLI) を使用したキャッシュの作成

Data Grid CLI を使用して、テンプレートから、または XML もしくは JSON 形式の設定ファイルでキャッシュを追加します。

#### 前提条件

ユーザーを作成し、少なくとも1つの Data Grid サーバーインスタンスを開始している。

#### 手順

1. Data Grid への CLI 接続を作成します。
2. **create cache** コマンドを使用して、キャッシュ定義を追加します。
  - **--file** オプションを使用して、XML または JSON ファイルからキャッシュ定義を追加します。

```
[//containers/default]> create cache --file=configuration.xml mycache
```

- **--template** オプションを使用して、テンプレートからキャッシュ定義を追加します。

```
[//containers/default]> create cache --template=org.infinispan.DIST_SYNC mycache
```

#### ヒント

**--template=** 引数の後に Tab キーを押して、利用可能なキャッシュテンプレートをリスト表示します。

3. **ls** コマンドを使用して、キャッシュが存在することを確認します。

```
[//containers/default]> ls caches  
mycache
```

4. **describe** コマンドを使用して、キャッシュ設定を取得します。

```
[//containers/default]> describe caches/mycache
```

#### 参照資料

- [Creating Data Grid CLI Connections](#)
- [Performing Cache Operations with the Data Grid CLI](#)

### 2.1.1. XML の設定

XML 形式の Data Grid 設定はスキーマに準拠し、以下を含める必要があります。

- `<infinispan>` ルート要素。
- `<cache-container>` 定義。

#### XML 設定のサンプル

```
<infinispan>
  <cache-container>
    <distributed-cache name="myCache" mode="SYNC">
      <encoding media-type="application/x-protostream"/>
      <memory max-count="1000000" when-full="REMOVE"/>
    </distributed-cache>
  </cache-container>
</infinispan>
```

### 2.1.2. JSON 設定

JSON 形式の Data Grid 設定:

- キャッシュ定義のみが必要です。
- XML 設定の構造に従う必要があります。
  - XML 要素は JSON オブジェクトになります。
  - XML 属性は JSON フィールドになります。

#### JSON 設定の例

```
{
  "distributed-cache": {
    "name": "myCache",
    "mode": "SYNC",
    "encoding": {
      "media-type": "application/x-protostream"
    },
    "memory": {
      "max-count": 1000000,
      "when-full": "REMOVE"
    }
  }
}
```

## 2.2. キャッシュエントリーの追加

データコンテナに **key:value** ペアのエントリーを作成します。

#### 前提条件

データを保存できる Data Grid キャッシュを作成している。

## 手順

1. Data Grid への CLI 接続を作成します。
2. 次のように、エントリーをキャッシュに追加します。
  - キャッシュのコンテキストで **put** コマンドを使用します。

```
[//containers/default/caches/mycache]> put hello world
```

- **put** コマンドで **--cache =** を使用します。

```
[//containers/default]> put --cache=mycache hello world
```

3. **get** コマンドを使用して、エントリーを確認します。

```
[//containers/default/caches/mycache]> get hello  
world
```

## 2.3. キャッシュのクリアとエントリーの削除

Data Grid CLI を使用してキャッシュからデータを削除します。

### 手順

1. Data Grid への CLI 接続を作成します。
2. 次のいずれかを行います。
  - **clearcache** コマンドを使用してすべてのエントリーを削除します。

```
[//containers/default]> clearcache mycache
```

- **remove** コマンドを使用して特定のエントリーを削除します。

```
[//containers/default]> remove --cache=mycache hello
```

## 2.4. キャッシュの削除

キャッシュをドロップしてキャッシュを削除し、キャッシュに含まれるすべてのデータを削除します。

### 手順

1. Data Grid への CLI 接続を作成します。
2. **drop** コマンドでキャッシュを削除します。

```
[//containers/default]> drop cache mycache
```

## 第3章 バッチ操作の実行

インタラクティブに、またはバッチファイルを使用して、グループで操作を処理します。

### 前提条件

- Data Grid クラスタを実行中である。

### 3.1. ファイルを使用したバッチ操作の実行

一連の操作を含むファイルを作成し、それらを Data Grid CLI に渡します。

#### 手順

1. 一連の操作を含むファイルを作成している。  
たとえば、**mybatch** という名前のキャッシュを作成する **batch** という名前のファイルを作成し、キャッシュに2つのエントリを追加して、CLI から切断します。

```
$ cat > batch<<EOF
create cache --template=org.infinispan.DIST_SYNC mybatch
put --cache=mybatch hello world
put --cache=mybatch hola mundo
disconnect
EOF
```

2. CLI を実行し、ファイルを入力として指定します。

```
$ bin/cli.sh -c localhost:11222 -f batch
```

3. 新しい Data Grid CLI 接続を作成し、**mybatch** を確認します。

```
[/containers/default]> ls caches
__protobuf_metadata
mybatch
[/containers/default]> ls caches/mybatch
hola
hello
[/containers/default]> disconnect
[disconnected]>
```



#### 注記

CLI バッチファイルは、システムプロパティの拡張をサポートします。**`\${property}`** 形式を使用する文字列は、**property** システムプロパティの値に置き換えられます。

### 3.2. インタラクティブなバッチ操作の実行

標準の入カストリーム `stdin` を使用して、バッチ操作をインタラクティブに実行します。

#### 手順

1. インタラクティブモードで Data Grid CLI を起動します。

```
$ bin/cli.sh -c localhost:11222 -f -
```



### 注記

**-c** フラグを使用しない場合は、**connect** コマンドを実行する必要があります。

```
$ bin/cli.sh -f -
connect
```

2. バッチ操作を実行します。以下に例を示します。

```
create cache --template=org.infinispan.DIST_SYNC mybatch
put --cache=mybatch hello world
put --cache=mybatch hola mundo
disconnect
quit
```

### ヒント

インタラクティブモードでコマンドを追加するには、**echo** を使用します。

以下の例は、**echo describe** を使用してクラスター情報を取得する方法を示しています。

```
$ echo describe|bin/cli.sh -c localhost:11222 -f -
{
  "name" : "default",
  "version" : "10.0.0-SNAPSHOT",
  "coordinator" : false,
  "cache_configuration_names" : [ "org.infinispan.REPL_ASYNC", "__protobuf_metadata",
  "org.infinispan.DIST_SYNC", "qcache", "org.infinispan.LOCAL", "dist_cache_01",
  "org.infinispan.INVALIDATION_SYNC", "org.infinispan.REPL_SYNC",
  "org.infinispan.SCATTERED_SYNC", "mycache", "org.infinispan.INVALIDATION_ASYNC",
  "mybatch", "org.infinispan.DIST_ASYNC" ],
  "cluster_name" : "cluster",
  "physical_addresses" : "[192.168.1.7:7800]",
  "coordinator_address" : "thundercat-34689",
  "cache_manager_status" : "RUNNING",
  "created_cache_count" : "4",
  "running_cache_count" : "4",
  "node_address" : "thundercat-47082",
  "cluster_members" : [ "thundercat-34689", "thundercat-47082" ],
  "cluster_members_physical_addresses" : [ "10.36.118.25:7801", "192.168.1.7:7800" ],
  "cluster_size" : 2,
  "defined_caches" : [ {
    "name" : "__protobuf_metadata",
    "started" : true
  }, {
    "name" : "mybatch",
    "started" : true
  } ]
}
```

## 第4章 DATA GRID CLI の設定

Data Grid CLI の設定プロパティを定義します。

### 4.1. DATA GRID CLI プロパティと永続ストレージの設定

Data Grid CLI の起動操作を設定し、永続ストレージの場所をカスタマイズします。

#### 前提条件

少なくとも1人の Data Grid ユーザーを作成している。

#### 手順

1. オプションで、次のいずれかの方法で Data Grid CLI ストレージディレクトリーへのカスタムパスを設定します。
  - **cli.dir** システムプロパティの使用:

```
$ bin/cli.sh -Dcli.dir=/path/to/cli/storage ...
```
  - **ISPN\_CLI\_DIR** 環境変数の使用:

```
export ISPN_CLI_DIR=/path/to/cli/storage
$ bin/cli.sh ...
```
2. **config set** コマンドを使用して、設定プロパティの値を設定します。
3. **config get** コマンドで設定プロパティを確認します。

#### ヒント

**help config** を実行して、使用可能な設定プロパティを確認し、使用例を取得します。

### 4.2. コマンドエイリアスの作成

Data Grid CLI コマンドのエイリアスを作成して、カスタムショートカットを定義します。

#### 手順

1. **alias <alias>=<command>** コマンドを使用してエイリアスを作成します。たとえば、**quit** コマンドのエイリアスとして **q** を設定します。

```
[//containers/default]> alias q=quit
```
2. **alias** コマンドを実行して、定義されたエイリアスを確認します。

```
[//containers/default]> alias
alias q='quit'
```
3. **unalias** コマンドを使用してエイリアスを削除します。以下に例を示します。

```
[//containers/default]> unalias q
```

### 4.3. DATA GRID SERVER 接続の信頼

SSL/TLS 証明書を使用して Data Grid Server への Data Grid CLI 接続を保護します。Data Grid Server の SSL ID としてキーストアを作成する場合、CLI はサーバー証明書を検証して ID を検証できます。

#### 前提条件

- Data Grid Server の SSL ID を設定している。
- 少なくとも1人の Data Grid ユーザーを作成している。

#### 手順

1. 次の例のように、サーバーキーストアの場所を指定します。

```
$ bin/cli.sh config set truststore /home/user/my-trust-store.jks
```

2. 必要に応じて、キーストアのパスワードを次のように定義します。

```
$ bin/cli.sh config set truststore-password secret
```

3. CLI 設定を確認します。

```
$ bin/cli.sh config get truststore
truststore=/home/user/my-trust-store.jks
```

```
$ bin/cli.sh config get truststore-password
truststore-password=secret
```

#### 参照

[Setting Up SSL Identities for Data Grid Server](#)

### 4.4. DATA GRID CLI ストレージディレクトリー

Data Grid CLI は、設定を次のデフォルトディレクトリーに保存します。

オペレーティングシステム	デフォルトパス
Linux/Unix	<b>\$HOME/.config/red_hat_data_grid</b>
Microsoft Windows	<b>%APPDATA%/Sun/Java/red_hat_data_grid</b>
Mac OS	<b>\$HOME/Library/Java/red_hat_data_grid</b>

このディレクトリーには以下のファイルが格納されています。

**cli.properties**

CLI 設定プロパティの値を格納します。

**aliases**

コマンドエイリアスを格納します。

**history**

CLI 履歴を保存します。



## 第5章 カウンターの操作

カウンターは、オブジェクトの数を記録するアトミック増減分操作を提供します。

### 前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

### 5.1. カウンターの作成

Data Grid CLI を使用して強力なカウンターと弱いカウンターを作成します。

#### 手順

1. Data Grid への CLI 接続を作成します。
2. 適切な引数を指定して **create counter** コマンドを実行します。
  - a. **my-weak-counter** を作成します。

```
[//containers/default]> create counter --concurrency-level=1 --initial-value=5 --storage=PERSISTENT --type=weak my-weak-counter
```

- b. **my-strong-counter** を作成します。

```
[//containers/default]> create counter --initial-value=3 --storage=PERSISTENT --type=strong my-strong-counter
```

3. 使用可能なカウンターをリスト表示します。

```
[//containers/default]> ls counters  
my-strong-counter  
my-weak-counter
```

4. カウンター設定を確認します。

- a. **my-weak-counter** について説明します。

```
[//containers/default]> describe counters/my-weak-counter  
  
{  
  "weak-counter":{  
    "initial-value":5,  
    "storage":"PERSISTENT",  
    "concurrency-level":1  
  }  
}
```

- b. **my-strong-counter** について説明します。

```
[//containers/default]> describe counters/my-strong-counter
```

```
{
  "strong-counter":{
    "initial-value":3,
    "storage":"PERSISTENT",
    "upper-bound":5
  }
}
```

## 5.2. カウンターへのデルタの追加

任意の値でカウンターに増分または減分を適用します。

### 手順

1. カウンターを選択します。

```
[//containers/default]> counter my-weak-counter
```

2. 現在のカウントをリスト表示します。

```
[//containers/default/counters/my-weak-counter]> ls
5
```

3. カウンター値を **2** 増やします。

```
[//containers/default/counters/my-weak-counter]> add --delta=2
```

4. カウンター値を **-4** 減らします。

```
[//containers/default/counters/my-weak-counter]> add --delta=-4
```



### 注記

強力なカウンターは、演算が適用された後に値を返します。 **--quiet = true** を使用して、戻り値を非表示にします。

たとえば、 **my-strong-counter]> add --delta = 3 --quiet = true**。

弱いカウンターは空の応答を返します。

## 第6章 PROTOBUF メタデータを使用したキャッシュのクエリー

Data Grid は、Protocol Buffer (Protobuf) を使用してキャッシュ内のデータを構造化し、クエリーを実行できるようにすることをサポートしています。

### 前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

### 6.1. メディアタイプの設定

さまざまなメディアタイプのキャッシュエントリーをエンコードして、要件に最適な形式でデータを保存します。

たとえば、次の手順は、**application/x-protostream** メディアタイプを設定する方法を示しています。

#### 手順

1. **qcache** という名前の分散キャッシュを追加し、メディアタイプを設定する Data Grid 設定ファイルを作成します。次に例を示します。

```
<infinispan>
  <cache-container>
    <distributed-cache name="qcache">
      <encoding>
        <key media-type="application/x-protostream"/>
        <value media-type="application/x-protostream"/>
      </encoding>
    </distributed-cache>
  </cache-container>
</infinispan>
```

2. **--file=** オプションを使用して、**pcache.xml** から **qcache** を作成します。

```
[//containers/default]> create cache --file=pcache.xml pcache
```

3. **pcache** を確認します。

```
[//containers/default]> ls caches
pcache
__protobuf_metadata
[//containers/default]> describe caches/pcache
{
  "distributed-cache" : {
    "mode" : "SYNC",
    "encoding" : {
      "key" : {
        "media-type" : "application/x-protostream"
      },
      "value" : {
        "media-type" : "application/x-protostream"
      }
    }
  }
}
```

```

    },
    "transaction" : {
      "mode" : "NONE"
    }
  }
}

```

4. **pcache** にエントリーを追加し、エンコードを確認します。

```

[//containers/default]> put --cache=pcache good morning
[//containers/default]> cd caches/pcache
[//containers/default/caches/pcache]> get good
{
  "_type" : "string",
  "_value" : "morning"
}

```

## 6.2. PROTOBUF スキーマの登録

Protobuf スキーマには、メッセージと呼ばれるデータ構造が **.proto** 定義ファイルに含まれています。

### 手順

1. 次のメッセージを含む **person.proto** という名前のスキーマファイルを作成します。

```

package org.infinispan.rest.search.entity;

message Address {
  required string street = 1;
  required string postCode = 2;
}

message PhoneNumber {
  required string number = 1;
}

message Person {
  optional int32 id = 1;
  required string name = 2;
  required string surname = 3;
  optional Address address = 4;
  repeated PhoneNumber phoneNumbers = 5;
  optional uint32 age = 6;
  enum Gender {
    MALE = 0;
    FEMALE = 1;
  }

  optional Gender gender = 7;
}

```

2. **person.proto** を登録します。

```

[//containers/default]> schema --upload=person.proto person.proto

```

3. **person.proto** を確認します。

```
[[containers/default]> cd caches/___protobuf_metadata
[[containers/default/caches/___protobuf_metadata]> ls
person.proto
[[containers/default/caches/___protobuf_metadata]> get person.proto
```

### 6.3. PROTOBUF スキーマを使用したキャッシュのクエリー

Data Grid は、JSON を Protobuf に自動的に変換するため、キャッシュエントリーを JSON 形式で読み書きし、Protobuf スキーマを使用してクエリーを実行できます。

たとえば、次の JSON ドキュメントについて考えてみます。

#### lukecage.json

```
{
  "_type": "org.infinispan.rest.search.entity.Person",
  "id": 2,
  "name": "Luke",
  "surname": "Cage",
  "gender": "MALE",
  "address": {"street": "38th St", "postCode": "NY 11221"},
  "phoneNumbers": [{"number": 4444}, {"number": 5555}]
}
```

#### jessicajones.json

```
{
  "_type": "org.infinispan.rest.search.entity.Person",
  "id": 1,
  "name": "Jessica",
  "surname": "Jones",
  "gender": "FEMALE",
  "address": {"street": "46th St", "postCode": "NY 10036"},
  "phoneNumbers": [{"number": 1111}, {"number": 2222}, {"number": 3333}]
}
```

#### matthewmurdock.json

```
{
  "_type": "org.infinispan.rest.search.entity.Person",
  "id": 3,
  "name": "Matthew",
  "surname": "Murdock",
  "gender": "MALE",
  "address": {"street": "57th St", "postCode": "NY 10019"},
  "phoneNumbers": []
}
```

前述の各 JSON ドキュメントには次が含まれます。

- JSON ドキュメントが対応する Protobuf メッセージを識別する **\_type** フィールド。

- **person.proto** スキーマのデータ型に対応するいくつかのフィールド。

## 手順

1. **pcache** キャッシュに移動します。

```
[/containers/default/caches]> cd pcache
```

2. 各 JSON ドキュメントをエンタリーとしてキャッシュに追加します。次に例を示します。

```
[/containers/default/caches/pcache]> put --encoding=application/json --file=jessicajones.json  
jessicajones  
[/containers/default/caches/pcache]> put --encoding=application/json --  
file=matthewmurdock.json matthewmurdock  
[/containers/default/caches/pcache]> put --encoding=application/json --file=lukecage.json  
lukecage
```

3. エンタリーが存在することを確認します。

```
[/containers/default/caches/pcache]> ls  
lukecage  
matthewmurdock  
jessicajones
```

4. キャッシュをクエリーして、性別データ型が **MALE** である Protobuf **Person** エンティティーからエンタリーを返します。

```
[/containers/default/caches/pcache]> query "from org.infinispan.rest.search.entity.Person p  
where p.gender = 'MALE'"  
{  
  "total_results" : 2,  
  "hits" : [ {  
    "hit" : {  
      "_type" : "org.infinispan.rest.search.entity.Person",  
      "id" : 2,  
      "name" : "Luke",  
      "surname" : "Cage",  
      "gender" : "MALE",  
      "address" : {  
        "street" : "38th St",  
        "postCode" : "NY 11221"  
      },  
      "phoneNumbers" : [ {  
        "number" : "4444"  
      }, {  
        "number" : "5555"  
      } ]  
    }, {  
      "hit" : {  
        "_type" : "org.infinispan.rest.search.entity.Person",  
        "id" : 3,  
        "name" : "Matthew",  
        "surname" : "Murdock",
```

```
    "gender" : "MALE",  
    "address" : {  
      "street" : "57th St",  
      "postCode" : "NY 10019"  
    }  
  }  
}]  
}
```

## 第7章 クロスサイトレプリケーション操作の実行

異なる場所で実行されている Data Grid クラスターは、データをバックアップするために相互に検出および通信できます。

### 前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

### 7.1. バックアップ場所のオフラインおよびオンライン化

バックアップ場所を手動でオフラインにし、オンラインに戻します。

#### 手順

1. Data Grid への CLI 接続を作成します。
2. **site status** コマンドを使用して、バックアップの場所がオンラインかオフラインかを確認します。

```
//containers/default> site status --cache=cacheName --site=NYC
```



#### 注記

**--site** はオプションの引数です。設定されていない場合、CLI はすべてのバックアップ場所を返します。

3. 次のようにバックアップ場所を管理します。
  - **bring-online** コマンドを使用して、バックアップの場所をオンラインにします。

```
//containers/default> site bring-online --cache=customers --site=NYC
```

- **take-offline** コマンドを使用して、バックアップの場所をオフラインにします。

```
//containers/default> site take-offline --cache=customers --site=NYC
```

詳細と例については、**help site** コマンドを実行してください。

### 7.2. バックアップ場所への状態のプッシュ

キャッシュの状態をリモートのバックアップ場所に転送します。

#### 手順

1. Data Grid への CLI 接続を作成します。
2. 次の例のように、**site** コマンドを使用して状態の転送をプッシュします。

```
//containers/default> site push-site-state --cache=cacheName --site=NYC
```



詳細と例については、**help site** コマンドを実行してください。

## 第8章 コマンドリファレンス

Data Grid CLI コマンドのマニュアルページを確認してください。

### ヒント

**help** コマンドを使用して、CLI セッションから直接マニュアルページにアクセスします。

たとえば、**get** コマンドのマニュアルページを表示するには、次の手順を実行します。

```
$ help get
```

### 8.1. ADD(1)

#### 8.1.1. 名前

**add** - 任意の値でカウンターに増分または減分を適用します。

#### 8.1.2. 概要

```
add ['OPTIONS'] ['COUNTER_NAME']
```

#### 8.1.3. オプション

```
--delta='nnn'
```

カウンター値を増加または減少させるデルタを設定します。デフォルトは **1** です。

```
-q, --quiet='[true|false]'
```

強力なカウンターの戻り値を非表示にします。デフォルトは **false** です。

#### 8.1.4. 例

```
add --delta=10 cnt_a
```

**cnt\_a** の値を **10** 増やします。

```
add --delta=-5 cnt_a
```

**cnt\_a** の値を **5** 減らします。

#### 8.1.5. 関連項目

cas(1)、reset(1)

### 8.2. ALIAS(1)

#### 8.2.1. 名前

**alias** - エイリアスを作成または表示します。

#### 8.2.2. 概要

```
alias ['ALIAS-NAME']='COMMAND']
```

### 8.2.3. 例

**alias q=quit**

**quit** コマンドのエイリアスとして **q** を作成します。

**alias**

定義されているすべてのエイリアスをリスト表示します。

### 8.2.4. 関連項目

config(1)、unalias(1)

## 8.3. CACHE(1)

### 8.3.1. 名前

cache - 後続のコマンドのデフォルトキャッシュを選択します。

### 8.3.2. 概要

cache ['CACHE\_NAME']

### 8.3.3. 例

**cache mycache**

**mycache** を選択します。 **cd caches/mycache** を使用してリソースツリーをナビゲートするのと同じです。

### 8.3.4. 関連項目

cd(1)、clear(1)、container(1)、get(1)、put(1)、remove(1)

## 8.4. CAS(1)

### 8.4.1. 名前

cas - 強力なカウンターで 'compare-and-swap' 操作を実行します。

### 8.4.2. 概要

cas ['OPTIONS'] ['COUNTER\_NAME']

### 8.4.3. オプション

**--expect='nnn'**

カウンターの期待値を指定します。

**--value='nnn'**

カウンターに新しい値を設定します。

**-q, --quiet='[true|false]'**

戻り値を非表示にします。デフォルトは false です。

#### 8.4.4. 例

**cas --expect=10 --value=20 cnt\_a**

現在の値が **10** の場合にのみ、**cnt\_a** の値を **20** に設定します。

#### 8.4.5. 関連項目

add(1)、cas(1)、reset(1)

### 8.5. CD(1)

#### 8.5.1. 名前

cd - サーバーリソースツリーをナビゲートします。

#### 8.5.2. 説明

**PATH** は、絶対パスまたは現在のリソースに対する相対パスです。../ は親リソースを指定します。

#### 8.5.3. 概要

cd['PATH']

#### 8.5.4. 例

**cd caches**

リソースツリーの **caches** パスに変更します。

#### 8.5.5. 関連項目

cache(1)、ls(1)、container(1)

### 8.6. CLEARCACHE(1)

#### 8.6.1. 名前

clearcache - キャッシュからすべてのエントリーを削除します。

#### 8.6.2. 概要

clearcache ['CACHE\_NAME']

#### 8.6.3. 例

**clearcache mycache**

**mycache** からすべてのエントリーを削除します。

#### 8.6.4. 関連項目

cache(1)、drop(1)、remove(1)

## 8.7. CONFIG(1)

### 8.7.1. 名前

config - CLI 設定プロパティを管理します。

### 8.7.2. 概要

**config**

**config set** 'name' 'value'

**config get** 'name'

### 8.7.3. 説明

CLI 設定プロパティを管理 (リスト、設定、取得) します。

### 8.7.4. コマンドの概要

**config**

設定されているすべての設定プロパティをリスト表示します。

**config set** 'name' ['value']

特定のプロパティの値を設定します。値を指定しない場合、プロパティは設定されません。

**config get** 'name'

特定のプロパティの値を取得します。

### 8.7.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

**-h, --help**

コマンドまたはサブコマンドのヘルプページを表示します。

### 8.7.6. プロパティ

**autoconnect-url**

起動時に CLI が自動的に接続する URL を指定します。

**autoexec**

起動時に実行する CLI バッチファイルのパスを指定します。

**trustall**

すべてのサーバー証明書を信頼するかどうかを指定します。値は **false**(デフォルト) および **true** です。

**truststore**

サーバー ID を検証する証明書チェーンを含むキーストアへのパスを定義します。

**truststore-password**

キーストアにアクセスするためのパスワードを指定します。

### 8.7.7. 例

**config set autoconnect-url <http://192.0.2.0:11222>**

CLI の起動時に、カスタム IP アドレスでサーバーに接続します。

**config get autoconnect-url**

**autoconnect-url** 設定プロパティの値を返します。

**config set autoexec /path/to/mybatchfile**

CLI の起動時に、"mybatchfile" という名前のバッチファイルを実行します。

**config set trustall true**

すべてのサーバー証明書を信頼します。

**config set truststore /home/user/my-trust-store.jks**

"my-trust-store.jks" という名前のキーストアのパスを指定します。

**config set truststore-password secret**

必要に応じて、キーストアのパスワードを設定します。

### 8.7.8. 関連項目

alias(1)、unalias(1)

## 8.8. CONNECT(1)

### 8.8.1. 名前

connect - 実行中の Data Grid サーバーに接続します。

### 8.8.2. 説明

デフォルトは <http://localhost:11222> で、認証が必要な場合は認証情報の入力を求められます。

### 8.8.3. 概要

```
connect ['OPTIONS'] ['SERVER_LOCATION']
```

### 8.8.4. オプション

**-u, --username='USERNAME'**

Data Grid サーバーで認証するユーザー名を指定します。

**-p, --password='PASSWORD'**

パスワードを指定します。

### 8.8.5. 例

**connect 127.0.0.1:11322 -u test -p changeme**

100 のポートオフセットとサンプルの認証情報を使用して、ローカルで実行されているサーバーに接続します。

### 8.8.6. 関連項目

disconnect(1)

## 8.9. CONTAINER(1)

### 8.9.1. 名前

container - 後続のコマンドを実行するためのコンテナを選択します。

### 8.9.2. 概要

container ['CONTAINER\_NAME']

### 8.9.3. 例

#### container default

デフォルトのコンテナを選択します。これは、**cd containers/default** を使用してリソースツリーをナビゲートするのと同じです。

### 8.9.4. 関連項目

cd(1)、clear(1)、container(1)、get(1)、put(1)、remove(1)

## 8.10. COUNTER(1)

### 8.10.1. 名前

counter - 後続のコマンドのデフォルトカウンターを選択します。

### 8.10.2. 概要

counter ['COUNTER\_NAME']

### 8.10.3. 例

#### counter cnt\_a

**cnt\_a** を選択します。**cd counters/cnt\_a** を使用してリソースツリーをナビゲートするのと同じです。

### 8.10.4. 関連項目

add(1)、cas(1)

## 8.11. CREATE(1)

### 8.11.1. 名前

create - Data Grid サーバーにキャッシュとカウンターを作成します。

### 8.11.2. 概要

create cache ['OPTIONS'] **CACHE\_NAME**

```
create counter ['OPTIONS'] COUNTER_NAME
```

### 8.11.3. キャッシュ作成のオプション

```
-f, --file='FILE'
```

JSON または XML 形式の設定ファイルを指定します。

```
-t, --template='TEMPLATE'
```

設定テンプレートを指定します。タブのオートコンプリートを使用して、使用可能なテンプレートを表示します。

```
-v, --volatile='[true|false]'
```

キャッシュが永続的であるか揮発性であるかを指定します。デフォルトは false です。

### 8.11.4. カウンター作成のオプション

```
-t, --type='[weak|strong]'
```

カウンターが弱いか強いかを指定します。

```
-s, --storage='[PERSISTENT|VOLATILE]'
```

カウンターが永続的であるか揮発性であるかを指定します。

```
-c, --concurrency-level='nnn'
```

カウンターの同時並行性レベルを設定します。

```
-i, --initial-value='nnn'
```

カウンターの初期値を設定します。

```
-l, --lower-bound='nnn'
```

強力なカウンターの下限を設定します。

```
-u, --upper-bound='nnn'
```

強力なカウンターの上限を設定します。

### 8.11.5. 例

```
create cache --template=org.infinispan.DIST_SYNC mycache
DIST_SYNC テンプレートから MyCache という名前のキャッシュを作成します。
```

```
create counter --initial-value=3 --storage=PERSISTENT --type=strong cnt_a
cnt_a という名前の強力なカウンターを作成します。
```

### 8.11.6. 関連項目

```
drop(1)
```

## 8.12. DESCRIBE(1)

### 8.12.1. 名前

describe - リソースに関する情報を表示します。

### 8.12.2. 概要

```
describe ['PATH']
```



### 8.12.3. 例

**describe //containers/default**

デフォルトのコンテナに関する情報を表示します。

**describe //containers/default/caches/mycache**

**mycache** キャッシュに関する情報を表示します。

**describe //containers/default/caches/mycache/k1**

**k1** キーに関する情報を表示します。

**describe //containers/default/counters/cnt1**

**cnt1** カウンターに関する情報を表示します。

### 8.12.4. 関連項目

cd(1)、ls(1)

## 8.13. DISCONNECT(1)

### 8.13.1. 名前

disconnect - Data Grid サーバーとの CLI セッションを終了します。

### 8.13.2. 概要

disconnect

### 8.13.3. 例

**disconnect**

現在の CLI セッションを終了します。

### 8.13.4. 関連項目

connect(1)

## 8.14. DROP(1)

### 8.14.1. 名前

drop - キャッシュとカウンターを削除します。

### 8.14.2. 概要

drop cache **CACHE\_NAME**

drop counter **COUNTER\_NAME**

### 8.14.3. 例

### drop cache mycache

**mycache** キャッシュを削除します。

### drop counter cnt\_a

**cnt\_a** カウンターを削除します。

#### 8.14.4. 関連項目

create(1)、clearcache(1)

## 8.15. ENCODING(1)

### 8.15.1. 名前

encoding - キャッシュエントリーのエンコーディングを表示および設定します。

### 8.15.2. 説明

キャッシュに対するputおよびget操作のデフォルトのエンコーディングを設定します。引数が指定されていない場合、**encoding** コマンドは現在のエンコーディングを表示します。

有効なエンコーディングでは、次のような標準の MIME タイプ (IANA メディアタイプ) の命名規則が使用されます。

- **text/plain**
- **application/json**
- **application/xml**
- **application/octet-stream**

### 8.15.3. 概要

encoding ['ENCODING']

### 8.15.4. 例

#### encoding application/json

エントリーを **application/json** としてエンコードするように、現在選択されているキャッシュを設定します。

### 8.15.5. 関連項目

get(1)、put(1)

## 8.16. GET(1)

### 8.16.1. 名前

get - キャッシュからエントリーを取得します。

### 8.16.2. 概要

**get** ['OPTIONS'] **KEY**

### 8.16.3. オプション

**-c, --cache='NAME'**

エントリーを取得するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

### 8.16.4. 例

**get hello -c mycache**

**mycache** から **hello** という名前のキーの値を取得します。

### 8.16.5. 関連項目

query(1)、put(1)

## 8.17. HELP(1)

### 8.17.1. 名前

help - コマンドのマニュアルページを出力します。

### 8.17.2. 概要

**help** ['COMMAND']

### 8.17.3. 例

**help get**

getコマンドのマニュアルページを出力します。

### 8.17.4. 関連項目

version(1)

## 8.18. LOGGING(1)

### 8.18.1. 名前

logging - Data Grid サーバーのランタイムロギング設定を検査および操作します。

### 8.18.2. 概要

**logging list-loggers**

**logging list-appenders**

**logging set** ['OPTIONS'] [**LOGGER\_NAME**]

## logging remove **LOGGER\_NAME**

### 8.18.3. ロギング設定のオプション

**-l, --level='OFF|TRACE|DEBUG|INFO|WARN|ERROR|ALL'**

特定のロガーのログレベルを指定します。

**-a, --appender='APPENDER'**

特定のロガーに設定するアペンダーを指定します。このオプションは、複数のアペンダーに対して繰り返すことができます。



#### 注記

ロガー名なしで **logging set** を呼び出すと、ルートロガーが変更されます。

### 8.18.4. 例

#### **logging list-loggers**

利用可能なすべてのロガーをリスト表示します。

#### **logging set --level=DEBUG --appenders=FILE org.infinispan**

**org.infinispan** ロガーのログレベルを **DEBUG** に設定し、**FILE** アペンダーを使用するように設定します。

## 8.19. LS(1)

### 8.19.1. 名前

**ls** - 現在のパスまたは特定のパスのリソースをリスト表示します。

### 8.19.2. 概要

**ls** ['PATH']

### 8.19.3. 例

#### **ls caches**

使用可能なキャッシュをリスト表示します。

**ls ../**

親リソースをリスト表示します。

### 8.19.4. 関連項目

**cd(1)**

## 8.20. MIGRATE(1)

### 8.20.1. 名前

**migrate** - Data Grid のあるバージョンから別のバージョンにデータを移行します。

## 8.20.2. 概要

`migrate cluster synchronize`

`migrate cluster disconnect`

## 8.20.3. 説明

あるバージョンの Data Grid から別のバージョンにデータを移行するには、**migrate** コマンドを使用します。

## 8.20.4. コマンドの概要

クラスターの移行

`migrate cluster synchronize`

ソースクラスターとターゲットクラスターの間でデータを同期します。

`migrate cluster disconnect`

ターゲットクラスターをソースクラスターから切断します。

## 8.20.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

`-h, --help`

コマンドまたはサブコマンドのヘルプページを表示します。

## 8.20.6. クラスター同期オプション

`-c, --cache='name'`

同期するキャッシュの名前。

`-b, --read-batch='num'`

バッチで処理するエントリーの量。デフォルトは10000です。

`-t, --threads='num'`

使用するスレッドの数。デフォルトはサーバーのコア数です。

## 8.20.7. クラスター切断オプション

`-c, --cache='name'`

ソースから切断するキャッシュの名前。

# 8.21. PATCH(1)

## 8.21.1. 名前

`patch` - サーバーパッチを管理します。

## 8.21.2. 説明

サーバーパッチをリスト表示、説明、インストール、ロールバック、および作成します。

パッチは、サーバーをアップグレードして問題を解決したり、新しい機能を追加したりするためのアーティファクトを含む zip アーカイブファイルです。パッチは、異なるバージョンの複数のサーバーインストールにターゲットバージョンを適用できます。

### 8.21.3. 概要

**patch ls**

**patch install** 'patch-file'

**patch describe** 'patch-file'

**patch rollback**

**patch create** 'patch-file' 'target-server' 'source-server-1' ['source-server-2'...]

### 8.21.4. パッチリストのオプション

**--server='path/to/server'**

現在のサーバーのホームディレクトリ外のターゲットサーバーへのパスを設定します。

**-v、--verbose**

個々のファイルに関する情報を含む、インストールされている各パッチの内容を表示します。

### 8.21.5. パッチインストールのオプション

**--dry-run**

パッチが変更を適用せずに実行する操作を示します。

**--server='path/to/server'**

現在のサーバーのホームディレクトリ外のターゲットサーバーへのパスを設定します。

### 8.21.6. パッチ説明のオプション

**-v、--verbose**

個々のファイルに関する情報を含む、パッチの内容を表示します。

### 8.21.7. パッチロールバックのオプション

**--dry-run**

パッチが変更を適用せずに実行する操作を示します。

**--server='path/to/server'**

現在のサーバーのホームディレクトリ外のターゲットサーバーへのパスを設定します。

### 8.21.8. パッチ作成のオプション

**-q, --qualifier='name'**

パッチの説明的な修飾子文字列を指定します (例:'one-off for issue nnnn')。

### 8.21.9. 例

**patch ls**

サーバーに現在インストールされているパッチをインストール順にリスト表示します。

**patch install mypatch.zip**

サーバーの現在のディレクトリーに"mypatch.zip"をインストールします。

**patch install mypatch.zip --server=/path/to/server/home**

サーバーの別のディレクトリーに"mypatch.zip"をインストールします。

**patch describe mypatch.zip**

"mypatch.zip"のターゲットバージョンとソースバージョンのリストを表示します。

**patch create mypatch.zip 'target-server' 'source-server-1' ['source-server-2'...]**

ターゲットサーバーのバージョンを使用し、ソースサーバーのバージョンに適用する"mypatch.zip"という名前のパッチファイルを作成します。

**patch rollback**

サーバーに適用された最後のパッチをロールバックし、以前のバージョンを復元します。

## 8.22. PUT(1)

### 8.22.1. 名前

put - キャッシュエントリーを追加または更新します。

### 8.22.2. 説明

新しいキーのエントリーを作成します。既存のキーの値を置き換えます。

### 8.22.3. 概要

put ['OPTIONS'] KEY [VALUE]

### 8.22.4. オプション

**-c, --cache='NAME'**

キャッシュの名前を指定します。デフォルトは現在選択されているキャッシュです。

**-e, --encoding='ENCODING'**

値のメディアタイプを設定します。

**-f, --file='FILE'**

エントリーの値を含むファイルを指定します。

**-l, --ttl='TTL'**

エントリーが自動的に削除されるまでの秒数 (存続時間) を設定します。0 の場合または指定されていない場合、デフォルトはキャッシュ設定の **lifespan** の値になります。負の値を設定すると、エントリーが削除されることはありません。

**-i, --max-idle='MAXIDLE'**

エントリーをアイドル状態にできる秒数を設定します。最大アイドル時間が経過してもエントリーの読み取りまたは書き込み操作が発生しない場合、エントリーは自動的に削除されます。0 の場合または指定されていない場合、デフォルトはキャッシュ設定の **maxidle** の値になります。負の値を設定すると、エントリーが削除されることはありません。

**-a, --if-absent=[true|false]**

エントリーが存在しない場合にのみエントリーを配置します。

### 8.22.5. 例

```
put -c mycache hello world
```

値が **world** の **hello** キーを **mycache** キャッシュに追加します。

```
put -c mycache -f myfile -i 500 hola
```

値が **myfile** の内容の **hola** キーを追加します。また、最大アイドル時間を **500** 秒に設定します。

### 8.22.6. 関連項目

get(1)、remove(1)

## 8.23. QUERY(1)

### 8.23.1. 名前

query - lckle クエリー文字列に一致するエントリーを取得します。

### 8.23.2. 概要

```
query ['OPTIONS'] QUERY_STRING
```

### 8.23.3. オプション

```
-c, --cache='NAME'
```

照会するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

```
--max-results='MAX_RESULTS'
```

返す結果の数を設定します。デフォルトは **10** です。

```
-o, --offset='OFFSET'
```

返される最初の結果のインデックスを指定します。デフォルトは **0** です。

```
--query-mode='QUERY_MODE'
```

サーバーがクエリーを実行する方法を指定します。値は **FETCH** と **BROADCAST** です。デフォルトは **FETCH** です。

### 8.23.4. 例

```
query "from org.infinispan.rest.search.entity.Person p where p.gender = 'MALE'"
```

現在選択されているキャッシュをクエリーして、性別データ型が **MALE** である Protobuf **Person** エンティティからエントリーを返します。

### 8.23.5. 関連項目

schema(1)

## 8.24. QUIT(1)

### 8.24.1. 名前



---

quit - コマンドラインインターフェイスを終了します。

### 8.24.2. 概要

quit

### 8.24.3. 例

quit

CLI を終了します。

### 8.24.4. 関連項目

disconnect(1)、shutdown(1)

## 8.25. REMOVE(1)

### 8.25.1. 名前

remove - キャッシュからエントリーを削除します。

### 8.25.2. 概要

remove **KEY** ['OPTIONS']

### 8.25.3. オプション

--cache='NAME'

エントリーを削除するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

### 8.25.4. 例

**remove --cache=mycache hola**

**mycache** キャッシュから **hola** エントリーを削除します。

### 8.25.5. 関連項目

cache(1)、drop(1)、clearcache(1)

## 8.26. RESET(1)

### 8.26.1. 名前

reset - カウンターの初期値を復元します。

### 8.26.2. 概要

reset ['COUNTER\_NAME']

### 8.26.3. 例

**reset cnt\_a**

**cnt\_a** カウンターをリセットします。

### 8.26.4. 関連項目

add(1)、cas(1)、drop(1)

## 8.27. SCHEMA(1)

### 8.27.1. 名前

schema - protobuf スキーマをアップロードして登録します。

### 8.27.2. 概要

schema ['OPTIONS'] **SCHEMA\_NAME**

### 8.27.3. オプション

**-u, --upload='FILE'**

指定された名前の protobuf スキーマとしてファイルをアップロードします。

### 8.27.4. 例

**schema --upload=person.proto person.proto**

Protobuf スキーマ **person.proto** を登録します。

### 8.27.5. 関連項目

query(1)

## 8.28. SHUTDOWN(1)

### 8.28.1. 名前

shutdown - 実行中のサーバーを停止するか、クラスターを正常に停止します。

### 8.28.2. 概要

shutdown server ['SERVERS']

shutdown cluster

### 8.28.3. 例

**shutdown server**

CLI が接続されているサーバーを停止します。

**shutdown server my\_server01**

ホスト名が **my\_server01** のサーバーを停止します。

**shutdown cluster**

クラスターの状態を保存し、キャッシュストアを使用する場合はエントリーを永続化し、すべてのノードを停止します。

**8.28.4. 関連項目**

connect(1)、disconnect(1)、quit(1)

**8.29. SITE(1)****8.29.1. 名前**

site - バックアップの場所を管理し、サイト間のレプリケーション操作を実行します。

**8.29.2. 概要**

site status ['OPTIONS']

site bring-online ['OPTIONS']

site take-offline ['OPTIONS']

site push-site-state ['OPTIONS']

site cancel-push-state ['OPTIONS']

site cancel-receive-state ['OPTIONS']

site push-site-status ['OPTIONS']

**8.29.3. オプション**

--cache='CACHE\_NAME'

キャッシュを指定します。

--site='SITE\_NAME'

バックアップの場所を指定します。

**8.29.4. 例**

**site status --cache=mycache**

**mycache** のすべてのバックアップ場所のステータスを返します。

**site status --cache=mycache --site=NYC**

**mycache** の **NYC** のステータスを返します。

**site bring-online --cache=mycache --site=NYC**

**mycache** のサイト **NYC** をオンラインにします。

**site take-offline --cache=mycache --site=NYC**

**mycache** のサイト **NYC** をオフラインにします。

**site push-site-state --cache=mycache --site=NYC**

キャッシュをリモートバックアップの場所にバックアップします。

**site push-site-status --cache=mycache**

**mycache** をバックアップする操作のステータスを表示します。

**site cancel-push-state --cache=mycache --site=NYC**

**mycache** を **NYC** にバックアップする操作をキャンセルします。

**site cancel-receive-state --cache=mycache --site=NYC**

**NYC** から状態を受信する操作をキャンセルします。

**site clear-push-state-status --cache=myCache**

**mycache** の状態をプッシュする操作のステータスをクリアします。

## 8.30. STATS(1)

### 8.30.1. 名前

**stats** - リソースに関する統計を表示します。

### 8.30.2. 概要

**stats** ['PATH']

### 8.30.3. 例

**stats //containers/default**

デフォルトのコンテナに関する統計を表示します。

**stats //containers/default/caches/mycache**

**mycache** キャッシュに関する統計を表示します。

### 8.30.4. 関連項目

[cd\(1\)](#)、[ls\(1\)](#)、[describe\(1\)](#)

## 8.31. TASK(1)

### 8.31.1. 名前

**task** - サーバー側のタスクとスクリプトを実行してアップロードします

### 8.31.2. 概要

**task upload** --file='script' 'TASK\_NAME'

**task exec** ['TASK\_NAME']

### 8.31.3. 例

**task upload --file=hello.js hello**

**hello.js** ファイルからスクリプトをアップロードし、**hello** という名前を付けます。

**task exec @@cache@names**

使用可能なキャッシュ名を返すタスクを実行します。

**task exec hello -Pgreetee=world**

**hello** という名前のスクリプトを実行し、**world** の値で **greetee** パラメーターを指定します。

### 8.31.4. オプション

**-P, --parameters='PARAMETERS'**

パラメーター値をタスクとスクリプトに渡します。

**-f, --file='FILE'**

指定された名前のスクリプトファイルをアップロードします。

### 8.31.5. 関連項目

ls(1)

## 8.32. UNALIAS(1)

### 8.32.1. 名前

unalias - エイリアスを削除します。

### 8.32.2. 概要

**unalias** 'ALIAS-NAME'

### 8.32.3. 例

**unalias q**

**q** エイリアスを削除します。

### 8.32.4. 関連項目

config(1)、alias(1)

## 8.33. USER(1)

### 8.33.1. 名前

user - プロパティセキュリティレルムで Data Grid ユーザーを管理します。

### 8.33.2. 概要

**user ls**

**user create** 'username'

```
user describe 'username'
```

```
user remove 'username'
```

```
user password 'username'
```

```
user groups 'username'
```

```
user encrypt-all
```

### 8.33.3. 説明

プロパティーセキュリティレームに保存されているユーザーを管理（リスト、作成、説明、削除、変更）します。注記：このコマンドは、プロパティーレームでのみ使用できます。

### 8.33.4. コマンドの概要

```
user ls
```

プロパティーファイルに存在するユーザーまたはグループをリスト表示します。

```
user create 'username'
```

パスワードの入力を求めた後、ユーザーを作成します。

```
user describe 'username'
```

ユーザー名、レーム、およびユーザーが属するグループを含め、ユーザーについて説明します。

```
user remove 'username'
```

指定されたユーザーをプロパティーファイルから削除します。

```
user password 'username'
```

ユーザーのパスワードを変更します。

```
user groups 'username'
```

ユーザーが属するグループを設定します。

```
user encrypt-all
```

プレーンテキストのユーザープロパティーファイル内のすべてのパスワードを暗号化します。

### 8.33.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

```
-h, --help
```

コマンドまたはサブコマンドのヘルプページを表示します。

```
-s, --server-root='path-to-server-root'
```

サーバールートへのパス。デフォルトは **server** です。

```
-f, --users-file='users.properties'
```

ユーザーパスワードを含むプロパティーファイルの名前。デフォルトは **users.properties** です。

```
-w, --groups-file='groups.properties'
```

ユーザーからグループへのマッピングを含むプロパティーファイルの名前。デフォルトは **groups.properties** です。

### 8.33.6. ユーザー作成/変更のオプション

**-a, --algorithms**

パスワードのハッシュに使用されるアルゴリズムを指定します。

**-g, --groups='group1,group2,...'**

ユーザーが属するグループを指定します。

**-p, --password='password'**

ユーザーのパスワードを指定します。

**-r, --realm='realm'**

レルム名を指定します。

**--plain-text**

パスワードをプレーンテキストで保存するかどうかを定義します (非推奨)。

### 8.33.7. ユーザーリストのオプション

**--groups**

ユーザーの代わりにグループのリストを表示します。

### 8.33.8. ユーザー暗号化 (すべて) のオプション

**-a, --algorithms**

パスワードのハッシュに使用されるアルゴリズムを指定します。

## 8.34. VERSION(1)

### 8.34.1. 名前

`version` - サーバーのバージョンと CLI のバージョンを表示します。

### 8.34.2. 概要

`version`

### 8.34.3. 例

**version**

サーバーと CLI のバージョンを返します。

### 8.34.4. 関連項目

`help(1)`